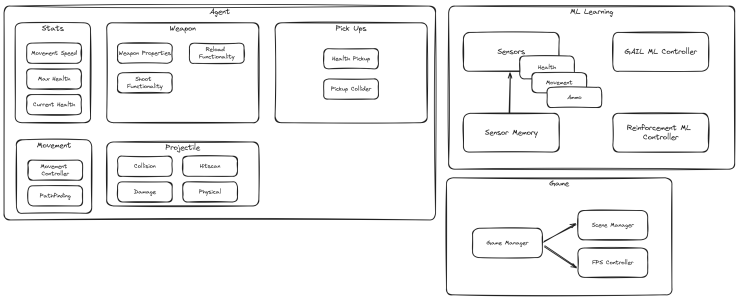
Project 2-1 ML Group 18

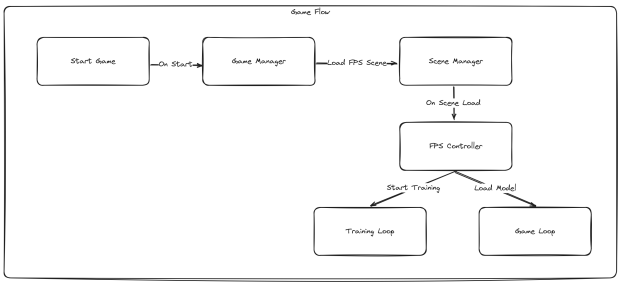
Introduction

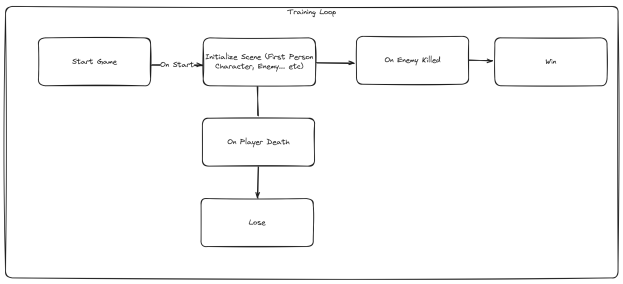
This project aims to develop a simplified 3D First-Person Shooter (FPS) game in which two AI-controlled agents compete against each other or compete against a human player.
The game will incorporate fundamental mechanics commonly found in FPS

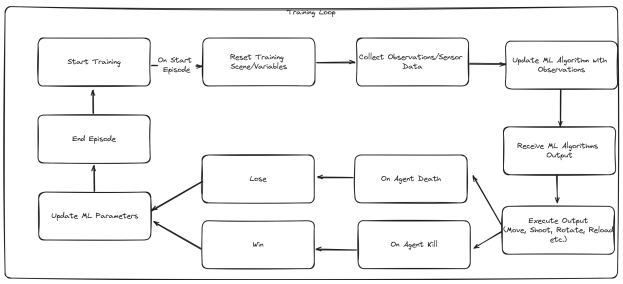
games, including guns, ammo, and health systems, providing a foundational exploration of autonomous agent behaviour in an interactive gaming environment.

Block Diagram









Software Component Preparation

Unity

The development of the game will be centered in the Unity editor, where the core scene setup, including visuals, colliders, GameObjects, and associated components, will be constructed. The use of `ScriptableObjects` will play a crucial role in defining data structures and configuring settings for various entities. For

instance, a `WeaponScriptableObject` may encapsulate attributes such as projectile type, damage output, and other relevant properties.

The use of `ScriptableObjects` will facilitate a data-driven design approach, where the data associated with entities is made accessible in the Unity Inspector. This approach enables designers to adjust and customize behaviours without modifying the underlying code, enhancing flexibility and iterative development.

C#

The core gameplay elements, including agents, sensors, pickups, managers, and controllers, will be implemented in C# using `MonoBehaviours` scripts to allow integration within the Unity Inspector.

Data definitions will be formalized through C# classes using `ScriptableObjects` scripts to expose the data to the Unity Inspector.

Additionally, custom logic for machine learning algorithms, such as Reinforcement Learning (RL), will be implemented in C#. This will include defining reward functions, setting criteria for the beginning and end of training episodes, and managing other critical aspects of the training process.

Python

Custom Python scripting is not planned, as the machine learning algorithms utilized in the project are natively supported by the ML-Agents package. However, Python remains integral to the training process, as it facilitates the execution of machine learning workflows. The ML-Agents package leverages Python's capabilities to orchestrate the training of agents, manage the communication between Unity and the training environment, and optimize learning parameters.

While the configuration and customization will primarily be handled within Unity and C#, Python will underpin the execution of the training algorithms to achieve the desired agent behaviours.

Task Description

For timeframe and who is assigned to each task please refer to the GitHub and Gantt Chart.

Phase 2: Core Functionality Implementation

The second phase of the project will focus on realizing the game's essential functionality, ensuring the core systems are fully implemented. This includes developing fundamental behaviours such as agent movement, agent statistics management, weapon mechanics (including shooting and reloading), as well as establishing win and lose conditions. These foundational elements will lay the groundwork for more advanced features and machine learning integration.

Upon completion of the core functionality, the project will progress to the integration of two machine learning algorithms: Generative Adversarial Imitation Learning (GAIL) and Reinforcement Learning (RL). The agents will be trained using these algorithms to enhance their behaviours and decision-making processes within the game environment.

Phase 3: Algorithm Optimization and Performance Profiling

In the third phase, the project we will refine the implemented machine learning algorithms. This will involve experimenting with the GAIL and RL algorithms, adjusting their parameters to optimize agent behaviour and performance.

Additionally, the performance of these algorithms will be profiled and analyzed to identify potential areas for improvement, ensuring that the agents' learning processes are both efficient and effective.

Phase 2

- Create Basic FPS Agent
    - Create a basic agent controller that will manage the logic for movement, shooting and stats.
- Create Agent Movement
    - Create movement controllers that will move and rotate an agent given input.
- Create Weapon System

    - Create a weapon with a visual model and behaviour for shooting projectiles and reloading ammo.
- Create Projectile System
    - Create physical and hit-scan projectiles with collision and damage.
- Create Health Pickups
    - Create an entity that agents can collide and pick up. Create a health pickup that will restore an agent's health when picked up.
- Create Agent Stats
    - Create health and movement speed stats for agents
- Create Agent Vision Sensor
    - Create a sensor that detects enemies within an agent's field of view. Records the entity detected and its position.
- Create Agent Sound Sensor
    - Create a sensor that detects noise produced from various actions such as moving, shooting and reloading. Records the estimated position and action of the sound.
- Create Agent Ammo Sensor
    - Create a sensor that detects the current ammo of an agent. Records the current and maximum ammo.
- Create Agent Health Sensor
    - Create a sensor that detects the current health of an agent. Records the current and maximum health.
- Create Agent Sensor Memory
    - Add memory to other sensors to maintain a history of recently sensed data.
- Create Game Flow
    - Create a class to manage the initialization of the game for the type of AI model to use, saving and loading AI Model and AI vs AI or AI vs Player

behaviour.

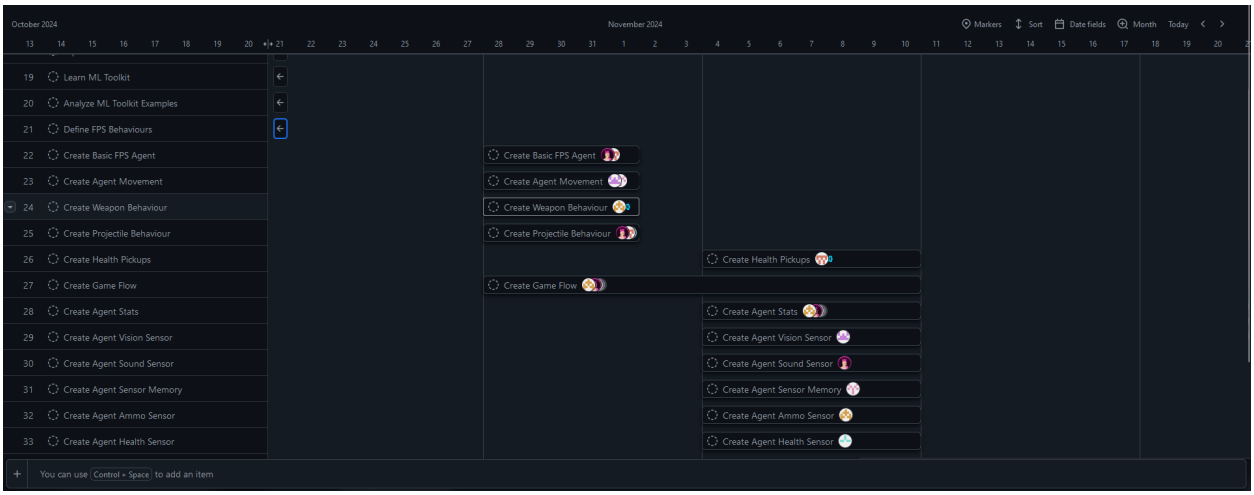
- Integrate Reinforcement Learning Algorithm
  - Integrate a Reinforcement Learning algorithm with Agent behaviour to record positive or negative actions an agent takes.
- Integrate GAIL ML Algorithm
  - Integrate a Generative Adversarial Imitation Learning algorithm. Define initial behaviour to imitate. Implement logic for cost function.

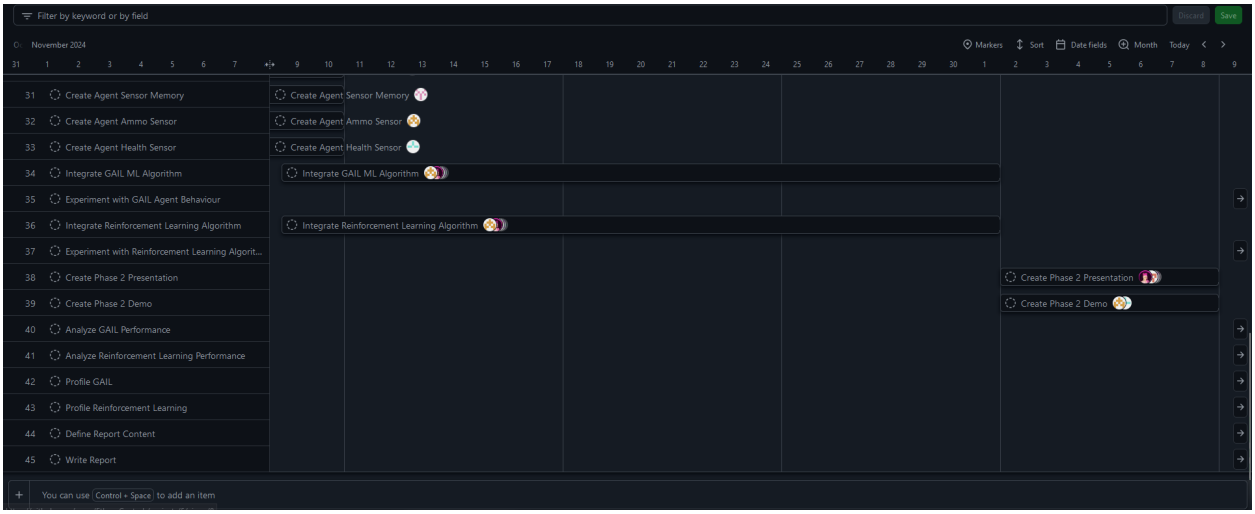
Phase 3

- Experiment with Machine Learning Algorithms
  - Tweak parameters in ML algorithms and analyze results on agents' behaviours.
- Profile GAIL
  - Analyze the performance and efficiency of the GAIL algorithm using Unity's in-built Profiler
- Profile Reinforcement Learning
  - Analyze the performance and efficiency of the Reinforcement Learning algorithm using Unity's in-built Profiler
- Define Report Content
  - Determine what we will include in the report
- Write Report
  - Write a report that outlines our whole project process from beginning to end and presents our findings.

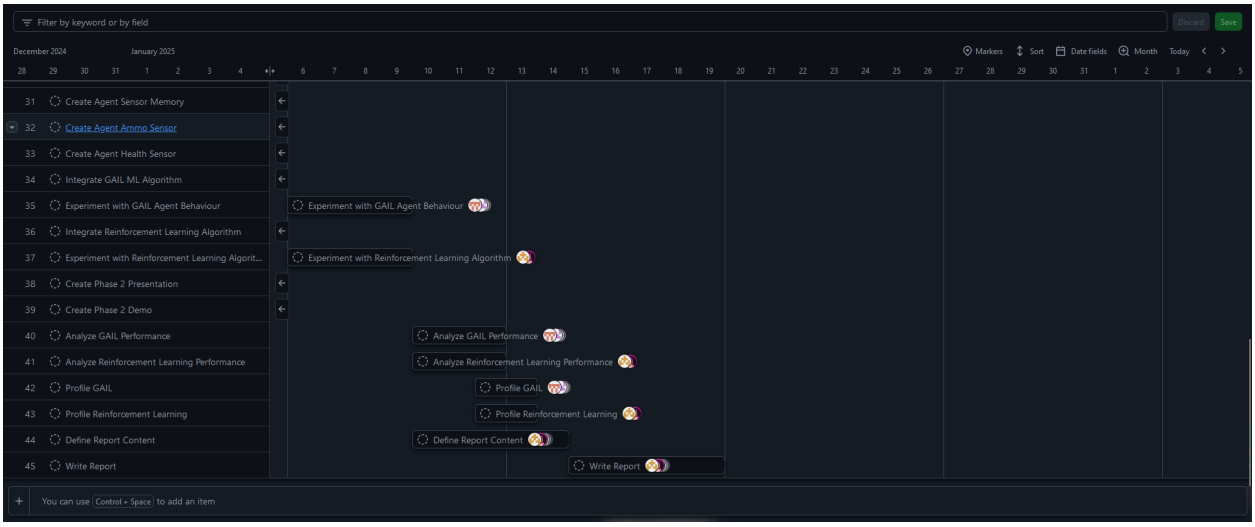
Gantt Chart

Phase 2


October 2024
November 2024
Markers
Sort
Date fields
Month
Today
19 Learn ML Toolkit
20 Analyze ML Toolkit Examples
21 Define FPS Behaviours
22 Create Basic FPS Agent
23 Create Agent Movement
24 Create Weapon Behaviour
25 Create Projectile Behaviour
26 Create Health Pickups
27 Create Game Flow
28 Create Agent Stats
29 Create Agent Vision Sensor
30 Create Agent Sound Sensor
31 Create Agent Sensor Memory
32 Create Agent Ammo Sensor
33 Create Agent Health Sensor
You can use Control + Space to add an item


Filter by keyword or by field
Discard
Save
November 2024
Markers
Sort
Date fields
Month
Today
31 Create Agent Sensor Memory
32 Create Agent Ammo Sensor
33 Create Agent Health Sensor
34 Integrate GAIL ML Algorithm
35 Experiment with GAIL Agent Behaviour
36 Integrate Reinforcement Learning Algorithm
37 Experiment with Reinforcement Learning Algorit...
38 Create Phase 2 Presentation
39 Create Phase 2 Demo
40 Analyze GAIL Performance
41 Analyze Reinforcement Learning Performance
42 Profile GAIL
43 Profile Reinforcement Learning
44 Define Report Content
45 Write Report
You can use Control + Space to add an item

Phase 3


Filter by keyword or by field
Discard
Save
December 2024
January 2025
Markers
Sort
Date fields
Month
Today
31 Create Agent Sensor Memory
32 Create Agent Ammo Sensor
33 Create Agent Health Sensor
34 Integrate GAIL ML Algorithm
35 Experiment with GAIL Agent Behaviour
36 Integrate Reinforcement Learning Algorithm
37 Experiment with Reinforcement Learning Algorit...
38 Create Phase 2 Presentation
39 Create Phase 2 Demo
40 Analyze GAIL Performance
41 Analyze Reinforcement Learning Performance
42 Profile GAIL
43 Profile Reinforcement Learning
44 Define Report Content
45 Write Report
You can use Control + Space to add an item

Risk Analysis

Custom Scene

Problem

Building our custom first person shooter scene comes with its own risks. One of them being that the development of the scene might take too long and further development be halted as a result of this.

Solution:

To solve this issue we will revert to the example scenes provided and continue development from there.

Custom Behaviour:

Problem:

Implementing the game functionalities such as agent movement and weapon behaviours might prove too challenging.

Solution:

To solve this issue we will simplify the behaviour, such as removing weapons and health, to an extent where it is easy enough to create the scene.

ML Implementation:

Problem:

Implementing custom behaviour for agents using RL and GAIL algorithms might prove too challenging. That integrating ML with the logic for movement and shooting is too difficult to implement within the given time frame or is too advanced.

Solution:

To solve this issue we will look for alternate ML algorithms that will simplify the integration process. If no alternative algorithms are found then this is not an issue with the ML Implementation but rather with the Custom Behaviour.

Optimized AI:

Problem:

Implementing ML to control Agents might prove too optimal. Such that an AI will always win against a Player.

Solution:

To solve this issue we will experiment and tweak the ML algorithm’s parameters.